UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/670,068	09/23/2003	Akihiko Tozawa	JP920020151US1	5679

67158 7590 05/28/2008

SHIMOKAJI & ASSOCIATES, P.C.
8911 RESEARCH DRIVE
IRVINE, CA 92618

EXAMINER	
PHAM, MICHAEL	

ART UNIT	PAPER NUMBER
2167	

NOTIFICATION DATE	DELIVERY MODE
05/28/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

patents@shimokaji.com
MSHIMOKAJI@SHIMOKAJI.COM

Office Action Summary	Application No.	Applicant(s)	
	10/670,068	TOZAWA ET AL.	
	Examiner	Art Unit	
	MICHAEL D. PHAM	2167	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 January 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3,4,6,10,11,14,15,18-20 and 22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3,4,6,10,11,14,15,18-20 and 22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☒ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Detailed Action

Status of claims

1. Claims 1, 3, 4, 6, 10, 11, 14-15, 18- 20, 22 are pending.

2. Claims 1, 3, 4, 6, 10, 11, 14-15, 18-20, 22 have been examined.

Specification

3. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

4. Prior objection to specifications is respectfully withdrawn.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claim 22 is rejected under 35 U.S.C. 102(b) as being anticipated by "Efficient Filtering of XML Documents for Selective Dissemination of Information", International Conference on Very Large Data Bases, 2000 by Altinel, Mehmet, and Franklin, Michael J. (hereafter Altinel).

Claim 22:

Atinel discloses the following claimed limitations:

"means for reading out an XPATH query automaton from a query automaton storage device and storing the XPATH query automaton, while reading in the document and performing a stream search by using states of a plurality of different types of nodes in the element identifiers included in the document and the query automaton;"[Figure 2, Figure 3a, and Figure 3b. Accordingly, means for reading out an XPATH query automaton from a query automaton storage device (figure 2, filter engine) and storing the XPATH query automaton (figure 2, query index), while reading in the document (figure 2, XML document) and performing a stream search (figure 3a and 3b, query) by using states of a plurality of different types of nodes (figure 3a and 3b, a, b, c, d, e) in the element identifiers (figure 3a and 3b, Q1, Q2, Q3, etc.) included in the document (figure 2, XML document) and the query automaton (figure 3b, query index)]

"means for identifying two different types of nodes included in said document, wherein states of a left node and a lower node in a tree structure are generated in correspondence with an identified element identifier;"[Accordingly, means for identifying two different types of nodes included in said document (figure 3a, Q1, a and b), wherein states of a left node (figure 3a, q1, a) and a lower node in a tree structure (figure 3a, q1, b) are generated in correspondence with an identified element identifier (Figure 3a and 3b, q1)]

"means for assigning a state transition among three states including a search state by using said identified input and a plurality of inputs registered in said query automaton node and"[page 57 col. 2, page 58 col. 1, and figure 3b. Accordingly, means for assigning a state

transition among three states (page 57 col. 2, current, initial, and final states) including a search state by using said identified input (page 58 col. 1, if it is not the final node, then the query is moved into its next state) and a plurality of inputs registered in said query automaton node (figure 3b)]

"a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node." [See page 57, column 2, 4th paragraph, All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List. The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph. If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)." Accordingly, a search result storage means for storing the output of the query automaton evaluator (A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List.), and thereafter outputting the stored output of the query automaton evaluator (candidate list) and the output of the searched node (candidate list)]

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1, 3-4, 6, 10, 11, 14-15, and 18-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Efficient Filtering of XML Documents for Selective Dissemination of Information", International Conference on Very Large Data Bases, 2000 by Altinel, Mehmet, and Franklin, Michael J. (hereafter Altinel) further in view of "XML Path Language (XPATH) 2.0", W3C Working Draft 16 August 2002, by W3C (hereafter W3C).

Claim 1:

Atinel discloses the following claimed limitations:

"a compiling device for generating an XPATH query automaton by storing an input query expression, performing parsing, identifying different types of nodes in said element identifiers," [Atinel discloses page 56 col. 2 Xfilter converts each XPATH query to a finite state machine. Atinenl further discloses Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Accordingly, a compiling device for generating an XPATH query automaton (Xfilter) by storing an input query expression (indexed queries), performing parsing (figure 3a, decomposition), identifying different types of nodes (figure 3a and 3b, a, b, c, d, e) in said element identifiers (figure 3a and 3b, query identifier) is disclosed.]

"wherein the compiling device generates and registers a state transition by replacing an XPATH axis including an XPATH axis in the opposite direction and a logical expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state;" [Atinel discloses page 56 col. 2, Xfilter converts each XPATH query to a finite state machine. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Further disclosing page 55 col. 1, how expressions such as //product [price/msrp < 300]/name are evaluated. Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Page 58 col. 1, discloses if it is not the final node of the query (i.e. final state) then the query is moved into its next state.

Accordingly, wherein the compiling device generates and registers a state transition by replacing an XPATH axis (Xfilter converts each XPATH query to a Finate State Machine) including an XPATH axis in the opposite direction (//product[price/msrp<300]/name], product) and a logical expression (//product[price/msrp<300]/name], price/msrp<300) while keeping an input query expression equal in terms of search (query index is used to match documents to individual XPATH queries) , wherein the compiling device generates a query automaton (figure 3, query index) including a plurality of states (query index is built over the states of the XPATH queries) of the backward nodes (figure 3, Q1, a), a condition for transition (price/msrp<300), and at least a search state (if it is not the final node, the query is moved into its next state), is disclosed.]

"A query automaton storage device storing the query automaton generated by said compiling device;" [page 55 col. 2, Xfilter is unique in that it combines the scalable SDI approach of indexing queries with the ability to reference document structure leaving to scalable but precise filtering of documents for internet-scale systems. Accordingly, a query automaton storage device storing the query automaton generated by said compiling device (indexing queries).]

"A query automaton evaluator for reading out said query automaton from said storage device storing said automaton" [Figure 2. a query automaton evaluator for reading out said query automaton (figure 2, filter engine) from said storage device storing said automaton (figure 2, query index).]

", while reading in said document and performing a stream search by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton" [figure 2 and page 55 col. 1, XPATH is used to select entire documents rather than parts of documents. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Accordingly, while reading in said document (figure 2, XML document) and performing a stream search (figure 3a and 3b, query) by using states (pg 56, states of the XPATH queries) of a plurality of different types of nodes (figures 3a and 3b, a, b, c, d, e) in said element identifiers (figures 3a and 3b, query identifiers) included in said document (XML document) and said query automaton (index query)]

", said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node,"[figure 2, figure 3a, page 56 col. 2, the query index is built over the states of the XPATH queries, and page 58, if this is the final path node of the query then the document is deemed to match the query. said query automaton evaluator (figure 2, filter engine) determining a state transition of a node (pg. 56, states of the XPATH query) under determination by storing a left node (figure 3a, q1, a) and a lower node (figure 3a, q1, b) in correspondence with an identified element identifier (figure 3a, q1), and evaluating said query automaton with a search result of said left node and said lower node (pg. 58, if this is the final path node of the query then the document is deemed to match the query.)]

"and outputting the searched node and a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node." [See page 57, column 2, 4th paragraph, All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List. The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph. If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)."

Accordingly, outputting the searched node (wait list) and a search result storage means for storing the output of the query automaton evaluator (candidate list), and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node (A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List)]

Altinel does not explicitly disclose, "a logical expression including a conjunction or a negative expression,".

On the other hand, W3C discloses on page 60, in addition to and- and or- expressions, Xpath provides a function named not that takes a general sequence as parameter and returns a Boolean value. Accordingly, a logical expression (logical expression) including a conjunction (AND) or a negative expression.

Both Altinel and W3C are within the same field of endeavor as Applicant's invention. It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to have applied the disclosure of W3C above to the disclosure of Altinel for the purpose of further evaluating and filtering documents.

Claim 3:

The combination of Altinel and W3C disclose in Altinel, "wherein said compiling device generates a query automaton with a state transition corresponding to an initial state, a final state,

and a search state registered thereon."[figure 2 and page 57 col. 2 initial state, final state, and current state. Accordingly, wherein said compiling device (Xfilter) generates a query automaton (page 56 col. 2 Xfilter converts each XPATH query to a finite state machine)with a state transition corresponding to an initial state (initial state), a final state (final state), and a search state (current state) registered thereon (index queries).]

Claim 4:

Altinel discloses the following claimed limitations:

"generating an XPATH query automaton by storing a query expression input by a compiling device, performing parsing, and identifying different types of nodes in said element identifiers," [Atinel discloses page 56 col. 2 Xfilter converts each XPATH query to a finite state machine. Atinenl further discloses Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Accordingly, generating an XPATH query automaton by storing an input query expression (indexed queries) by a compiling device (xfilter), performing parsing (figure 3a, decomposition), identifying different types of nodes (figure 3a and 3b, a, b, c, d, e) in said element identifiers (figure 3a and 3b, query identifier) is disclosed]

"by the steps of generating and registering a state transition by replacing an XPATH axis including an XPATH axis in the opposite direction and a logical expression while keeping an input query expression equal in terms of search, the query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state" [Atinel discloses page 56 col. 2, Xfilter converts each XPATH query to a finite state machine. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Further

disclosing page 55 col. 1, how expressions such as //product [price/msrp < 300]/name are evaluated. Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Page 58 col. 1, discloses if it is not the final node of the query (i.e. final state) then the query is moved into its next state.

Accordingly, by the steps of generating and registering a state transition by replacing an XPATH axis (Xfilter converts each XPATH query to a Finate State Machine) including an XPATH axis in the opposite direction (//product[price/msrp<300]/name], product) and a logical expression (//product[price/msrp<300]/name], price/msrp<300) while keeping an input query expression equal in terms of search (query index is used to match documents to individual XPATH queries) , the query automaton (figure 3, query index) including a plurality of states (query index is built over the states of the XPATH queries) of the backward nodes (figure 3, Q1, a), a condition for transition (price/msrp<300), and at least a search state (if it is not the final node, the query is moved into its next state), is disclosed.]

"said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node;"[figure 2, figure 3a, and page 58. Accordingly, said query automaton evaluator (figure 2, filter engine) determining a state transition of a node (page 56, states of the XPATH query) under determination by storing a left node (figure 3a, q1, a) and a lower node (figure 3a, q1, b) in correspondence with an identified element identifier (figure 3a, q1), and evaluating said query automaton with a search result of said left node and said lower node (pg. 58, if this is the final path node of the query then the document is deemed to match the query.)]

"storing the xpath query automaton generated by said compiling device in a query automaton storage device; and" [page 55 col. 2, Xfilter is unique in that it combines the scalable SDI approach of indexing queries with the ability to reference document structure leaving to scalable but precise filtering of documents for internet-scale systems. Accordingly, storing the xpath query automaton generated by said compiling device in a query automaton storage device (indexing queries).]

"reading out said XPATH query automaton from said query automaton storage device and storing said query automaton, while reading in said document and performing a stream search with a query automaton evaluator by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton, and"[figure 2 and page 55 col. 1, XPATH is used to select entire documents rather than parts of documents. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Accordingly, reading out said XPATH query automaton from said query automaton storage device (figure 2, query index) and storing said query automaton (figure 3b, query index), while reading in said document (figure 2, XML document) and performing a stream search (figure 3a and 3b, query) by using states (pg 56, states of the XPATH queries) of a plurality of different types of nodes (figures 3a and 3b, a, b, c, d, e) in said element identifiers (figures 3a and 3b, query identifiers) included in said document (XML document) and said query automaton (index query)]

"storing the output of the query automaton evaluator in a search result storage means for, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node."[See page 57, column 2, 4th paragraph, All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List. The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph. If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)." Accordingly, storing the output of the query automaton evaluator in a search result storage means for (A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List.), and thereafter outputting the stored output of the query automaton evaluator (candidate list) and the output of the searched node (candidate list)]

Altinel does not explicitly disclose, "a logical expression including a conjunction or a negative expression,".

On the other hand, W3C discloses on page 60, in addition to and- and or- expressions, Xpath provides a function named not that takes a general sequence as parameter and returns a

Boolean value. Accordingly, a logical expression (logical expression) including a conjunction (AND) or a negative expression.

Both Altinel and W3C are within the same field of endeavor as Applicant's invention. It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to have applied the disclosure of W3C above to the disclosure of Altinel for the purpose of further evaluating and filtering documents.

Claim 6:

The combination of Altinel and W3C disclose in Altinel, "wherein said step of generating an XPATH query automaton comprises a step of generating an XPATH query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon." [wherein said step of generating an XPATH query automaton comprises a step of generating an XPATH query automaton with a state transition (page 56 col. 2 Xfilter converts each XPATH query to a finite state machine)corresponding to an initial state (initial), a final state (final), and a search state (current state) registered thereon.]

Claim 10:

Altinel discloses the following claimed limitations:

"functioning as a compiling device for generating an XPATH query automaton by storing an input query expression, performing parsing, and identifying different types of nodes in said

element identifiers," [Atinel discloses page 56 col. 2 Xfilter converts each XPATH query to a finite state machine. Atinenl further discloses Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Accordingly, functioning as a compiling device (xfilter) for generating an XPATH query automaton by storing an input query expression (indexed queries), performing parsing (figure 3a, decomposition), identifying different types of nodes (figure 3a and 3b, a, b, c, d, e) in said element identifiers (figure 3a and 3b, query identifier) is disclosed]

"by the steps of generating and registering a state transition by replacing an XPATH axis including an XPATH axis in the opposite direction and a logical expression while keeping an input query expression equal in terms of search, the query automaton including a plurality of states of the backwards nodes, a condition for transition, and at least a search state," [Atinel discloses page 56 col. 2, Xfilter converts each XPATH query to a finite state machine. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Further disclosing page 55 col. 1, how expressions such as //product [price/msrp < 300]/name are evaluated. Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Page 58 col. 1, discloses if it is not the final node of the query (i.e. final state) then the query is moved into its next state. Accordingly, by the steps of generating and registering a state transition by replacing an XPATH axis (Xfilter converts each XPATH query to a Finate State Machine) including an XPATH axis in the opposite direction (//product[price/msrp<300]/name], product) and a logical expression (//product[price/msrp<300]/name], price/msrp<300) while keeping an input query expression

equal in terms of search (query index is used to match documents to individual XPATH queries) , the query automaton (figure 3, query index) including a plurality of states (query index is built over the states of the XPATH queries) of the backward nodes (figure 3, Q1, a), a condition for transition (price/msrp<300), and at least a search state (if it is not the final node, the query is moved into its next state), is disclosed.]

"said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node;" [figure 2, figure 3a, and page 58. Accordingly, said query automaton evaluator (figure 2, filter engine) determining a state transition of a node (page 56, states of the XPATH query) under determination by storing a left node (figure 3a, q1, a) and a lower node (figure 3a, q1, b) in correspondence with an identified element identifier (figure 3a, q1), and evaluating said query automaton with a search result of said left node and said lower node (pg. 58, if this is the final path node of the query then the document is deemed to match the query.)]

"storing a query automaton generated by said compiling device in a query automaton storage device;"[page 55 col. 2, Xfilter is unique in that it combines the scalable SDI approach of indexing queries with the ability to reference document structure leaving to scalable but precise filtering of documents for internet-scale systems. Accordingly, storing a query automaton generated by said compiling device in a query automaton storage device (indexing queries).]

"functioning as a query automaton evaluator for reading out said query automaton from said storage device and storing said query automaton, while reading in said document and performing a stream search by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton and"[figure 2 and page 55 col. 1, XPATH is used to select entire documents rather than parts of documents. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Accordingly, functioning as a query automaton evaluator for reading out said XPATH query automaton from said storage device (figure 2, query index) and storing said query automaton (figure 3b, query index), while reading in said document (figure 2, XML document) and performing a stream search (figure 3a and 3b, query) by using states (pg 56, states of the XPATH queries) of a plurality of different types of nodes (figures 3a and 3b, a, b, c, d, e) in said element identifiers (figures 3a and 3b, query identifiers) included in said document (XML document) and said query automaton (index query)]

"storing the output of the query automaton evaluator in a search result storage means, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node" [See page 57, column 2, 4th paragraph, All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List. The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph. If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not

the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)." Accordingly, storing the output of the query automaton evaluator in a search result storage means for (A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List.), and thereafter outputting the stored output of the query automaton evaluator (candidate list) and the output of the searched node (candidate list)]

Altinel does not explicitly disclose, "a logical expression including a conjunction or a negative expression,".

On the other hand, W3C discloses on page 60, in addition to and- and or- expressions, Xpath provides a function named not that takes a general sequence as parameter and returns a Boolean value. Accordingly, a logical expression (logical expression) including a conjunction (AND) or a negative expression.

Both Altinel and W3C are within the same field of endeavor as Applicant's invention. It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to have applied the disclosure of W3C above to the disclosure of Altinel for the purpose of further evaluating and filtering documents.

Claim 11:

The combination of Altinel and W3C disclose in Altinel, "wherein said performance of a stream search determines a state transition of a node under determination at the moment by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node, and wherein said query automaton is generated as a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon" [pg. 56 col. 2, figure 3a, and page 58. Accordingly, wherein said performance of a stream search determines a state transition of a node (page 56 col. 2, the query index is built over the states of the XPATH queries.) under determination at the moment by storing a left node (figure 3a, q1, a) and a lower node (figure 3a, q1, b) in correspondence with an identified element identifier (figure 3a, q1), and evaluating said query automaton with a search result of said left node and said lower node (pg. 58, if this is the final path node of the query then the document is deemed to match the query), and wherein said query automaton is generated as a query automaton with a state transition (page 56 col. 2 Xfilter converts each XPATH query to a finite state machine)corresponding to an initial state (initial), a final state (final), and a search state (current state) registered thereon.]

Claim 14:

Altinel discloses the following claimed limitations:

"generating and registering a state transition by replacing an XPATH axis including an XPATH axis in the opposite direction and a logical expression while keeping an input query expression equal in terms of search, and storing a plurality of states of a backward node in correspondence with said backward node into a query automaton storage device;" [Atinel

discloses page 56 col. 2, Xfilter converts each XPATH query to a finite state machine. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Further disclosing page 55 col. 1, how expressions such as //product [price/msrp < 300]/name are evaluated. Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Page 58 col. 1, discloses if it is not the final node of the query (i.e. final state) then the query is moved into its next state. Accordingly, generating and registering a state transition by replacing an XPATH axis (Xfilter converts each XPATH query to a Finate State Machine) including an XPATH axis in the opposite direction (//product[price/msrp<300]/name], product) and a logical expression (//product[price/msrp<300]/name], price/msrp<300) while keeping an input query expression equal in terms of search (query index is used to match documents to individual XPATH queries) , and storing a plurality of states of a backward node in correspondence with said backward node into a query automaton storage device (pg. 57 col. 2, candidate list), is disclosed.]

"generating a query automaton by registering a plurality of states of said backward node, a condition for transition, at least a search state, a reached state in correspondence with each other in said storage device, performing a parsing, and identifying different types of nodes in said element identifiers, by the steps of generating and registering a state transition by the query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state, said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and

said lower node and"[Atinel discloses page 56 col. 2, Xfilter converts each XPATH query to a finite state machine. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Further disclosing page 55 col. 1, how expressions such as //product [price/msrp < 300]/name are evaluated. Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Page 58 col. 1, discloses if it is not the final node of the query (i.e. final state) then the query is moved into its next state. Accordingly, generating a query automaton by registering a plurality of states of said backward node (figure 3b and pg. 56 query index is built over the states of the XPATH queries), a condition for transition (pg. 55, price/msrp<300), at least a search state (if it is not the final node, the query is moved into its next state), a reached state in correspondence with each other in said storage device (final node), performing a parsing (figure 3a, decomposition), and identifying different types of nodes (figure 3a and 3b, a, b, c, d, e) in said element identifiers (figure 3a and 3b, query identifier), by the steps of generating and registering a state transition by the query automaton including a plurality of states of the backward nodes (figure 3a and 3b, q1, a and b), a condition for transition (pg. 55, price/msrp<300), and at least a search state (if it is not the final node, the query is moved into its next state), said query automaton evaluator determining a state transition of a node under determination (figure 2, filter engine) by storing a left node and a lower node in correspondence with an identified element identifier (figure 3b, query index), and evaluating said query automaton with a search result of said left node and said lower node (pg. 58, if this is the final path node of the query then the document is deemed to match the query)is disclosed.]

"storing the output of the query automaton evaluator in a search result storage means, and thereafter outputting the stored output of the query automaton evaluator and the output of the search node." [See page 57, column 2, 4th paragraph, All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List. The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph. If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)." Accordingly, storing the output of the query automaton evaluator in a search result storage means for (A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List.), and thereafter outputting the stored output of the query automaton evaluator (candidate list) and the output of the searched node (candidate list)]

Altinel does not explicitly disclose, "a logical expression including a conjunction or a negative expression,".

On the other hand, W3C discloses on page 60, in addition to and- and or- expressions, Xpath provides a function named not that takes a general sequence as parameter and returns a

Boolean value. Accordingly, a logical expression (logical expression) including a conjunction (AND) or a negative expression.

Both Altinel and W3C are within the same field of endeavor as Applicant's invention. It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to have applied the disclosure of W3C above to the disclosure of Altinel for the purpose of further evaluating and filtering documents.

Claim 15:

The combination of Atinel and W3C disclose in Atinel "wherein said compiling method comprises a step of identifying said backward node as a left node or a lower node according to a type of said element identifier, and wherein said plurality of states are states of said left node and said lower node"[figure 3a and 3b, and page 57 col. 2 initial states, future states, final states, and current state. Accordingly, wherein said compiling method comprises a step of identifying said backward node as a left node or a lower node (figures 3a, q1, a or b) according to a type of said element identifier (figure 3a, q1, query id), and wherein said plurality of states are states of said left node and said lower node (pg. 57 col. 2, initial states and future states)].

Claim 18:

Atinel discloses the following claimed limitations:

"generating and registering a state transition by replacing an XPATH axis including an XPATH axis in the opposite direction and a logical expression while keeping an input query expression equal in term of search, and storing the plurality of states of a backward node in correspondence with said backward node into a storage device;" [Atinel discloses page 56 col. 2, Xfilter converts each XPATH query to a finite state machine. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Further disclosing page 55 col. 1, how expressions such as //product [price/msrp < 300]/name are evaluated. Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Page 58 col. 1, discloses if it is not the final node of the query (i.e. final state) then the query is moved into its next state.

Accordingly, generating and registering a state transition by replacing an XPATH axis (Xfilter converts each XPATH query to a Finate State Machine) including an XPATH axis in the opposite direction (//product[price/msrp<300]/name], product) and a logical expression (//product[price/msrp<300]/name], price/msrp<300) while keeping an input query expression equal in terms of search (query index is used to match documents to individual XPATH queries) , and storing a plurality of states of a backward node in correspondence with said backward node into a query automaton storage device (pg. 57 col. 2, candidate list), is disclosed.]

"generating a query automaton by registering a plurality of states of said backward node, a condition for transition, at least a search state, and a reached state in correspondence with each other in said storage device, performing parsing, and identifying different types of nodes in said element identifiers," [Atinel discloses page 56 col. 2, Xfilter converts each XPATH query to a

finite state machine. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Further disclosing page 55 col. 1, how expressions such as //product [price/msrp < 300]/name are evaluated. Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Page 58 col. 1, discloses if it is not the final node of the query (i.e. final state) then the query is moved into its next state.

Accordingly, generating a query automaton by registering a plurality of states of said backward node(figure 3b and pg. 56 query index is built over the states of the XPATH queries), a condition for transition(pg. 55, price/msrp<300), at least a search state (if it is not the final node, the query is moved into its next state), and a reached state in correspondence with each other in said storage device(final node), performing parsing(figure 3a, decomposition), and identifying different types of nodes figure 3a and 3b, a, b, c, d, e) in said element identifiers (figure 3a and 3b, query identifier)]

"by the steps of generating and registering a state transition by replacing an XPATH axis including an XPATH axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, the query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state, said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node and"[Atinel discloses page 56 col. 2, Xfilter converts each XPATH query to a finite state machine. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Further disclosing page 55 col. 1,

how expressions such as //product [price/msrp < 300]/name are evaluated. Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Page 58 col. 1, discloses if it is not the final node of the query (i.e. final state) then the query is moved into its next state.

Accordingly, by the steps of generating and registering a state transition by replacing an XPATH axis (Xfilter converts each XPATH query to a Finate State Machine) including an XPATH axis in the opposite direction (//product[price/msrp<300]/name], product) and a logical expression (//product[price/msrp<300]/name], price/msrp<300) while keeping an input query expression equal in terms of search (query index is used to match documents to individual XPATH queries) , the query automaton (figure 3, query index) including a plurality of states (query index is built over the states of the XPATH queries) of the backward nodes (figure 3, Q1, a), a condition for transition (price/msrp<300), and at least a search state (if it is not the final node, the query is moved into its next state), is disclosed.]

"said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node" [figure 2, figure 3b, and page 58. Accordingly, said query automaton evaluator determining a state transition of a node under determination (figure 2, filter engine) by storing a left node and a lower node in correspondence with an identified element identifier (figure 3b, query index), and evaluating said query automaton with a search result of said left node and said

lower node (pg. 58, if this is the final path node of the query then the document is deemed to match the query)is disclosed.]

"storing the output of the XPATH query automaton evaluator in a search result storage means, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node." [See page 57, column 2, 4th paragraph, All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List. The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph. If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)." Accordingly, storing the output of the XPATH query automaton evaluator in a search result storage means, (A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List.), and thereafter outputting the stored output of the query automaton evaluator (candidate list) and the output of the searched node (candidate list)]

Altinel does not explicitly disclose, "a logical expression including a conjunction or a negative expression,".

On the other hand, W3C discloses on page 60, in addition to and- and or- expressions, Xpath provides a function named not that takes a general sequence as parameter and returns a Boolean value. Accordingly, a logical expression (logical expression) including a conjunction (AND) or a negative expression.

Both Altinel and W3C are within the same field of endeavor as Applicant's invention. It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to have applied the disclosure of W3C above to the disclosure of Altinel for the purpose of further evaluating and filtering documents.

Claim 19:

The combination of Atinel and W3C disclose in Atinel "wherein said program comprises a step of causing a computer to identify said backward node as a left node or a lower node according to a type of said element identifier, and wherein said plurality of states are states of said left node and said lower node"[figure 3a and 3b, and page 57 col. 2 initial states, future states, final states, and current state. Accordingly, wherein said compiling method comprises a step of identifying said backward node as a left node or a lower node (figures 3a, q1, a or b) according to a type of said element identifier (figure 3a, q1, query id), and wherein said plurality of states are states of said left node and said lower node (pg. 57 col. 2, initial states and future states)].

Claim 20:

Atinel discloses the following claimed limitations:

"a compiling device for generating a two-state query automaton for enabling a state transition by storing an input query expression, performing parsing, and reading at least two states assigned to different types of nodes in said element identifiers," [Atinel discloses page 56 col. 2 Xfilter converts each XPATH query to a finite state machine. Atinenl further discloses Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Accordingly, a compiling device (Xfilter)for generating a two-state query automaton (figure 3a and 3b) for enabling a state transition by storing an input query expression (figure 3b), performing parsing (figure 3a, decomposition), and reading at least two states assigned to different types of nodes (figure 3a and 3b, a, b, c, d, e) in said element identifiers (figure 3a, query identifier) is disclosed.]

"wherein the compiling device generates and registers a state transition by replacing an XPATH axis including an XPATH axis in the opposite direction and a logical expression while keeping an input query expression equal in terms of search," [Atinel discloses page 56 col. 2, Xfilter converts each XPATH query to a finite state machine. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Further disclosing page 55 col. 1, how expressions such as //product [price/msrp < 300]/name are evaluated. Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Page 58 col. 1, discloses if it is not the final node of the query (i.e. final state) then the query is moved into its next state.

Accordingly, wherein the compiling device generates and registers a state transition by replacing an XPATH axis (Xfilter converts each XPATH query to a Finate State Machine) including an XPATH axis in the opposite direction (//product[price/msrp<300]/name], product)

and a logical expression while keeping an input query expression equal in terms of search (query index is used to match documents to individual XPATH queries), wherein the compiling device generates a query automaton (figure 3, query index) including a plurality of states (query index is built over the states of the XPATH queries) of the backward nodes (figure 3, Q1, a), a condition for transition (price/msrp<300), and at least a search state (if it is not the final node, the query is moved into its next state), is disclosed.]

"and wherein said two states are states of a left node and a lower node of a tree structure generated in correspondence with an identified element identifier and wherein said two input automaton uses three states of said automaton-evaluating device;"[figure 2, figure 3a, and page 57 col. 2. Accordingly, wherein said two states (figure 3a and 3b, q1, a and b) are states of a left node (figure 3a and 3b, q1, a) and a lower node (figure 3a and 3b, q1, b) of a tree structure generated in correspondence with an identified element identifier (figure 3a and 3b, q1) and wherein said two input automaton uses three states of said automaton-evaluating device (page 57 col. 2, initial state, final state, and current state.).]

"an automaton query storage device for storing said two-state input automaton;" [page 55 col. 2, Xfilter is unique in that it combines the scalable SDI approach of indexing queries with the ability to reference document structure leaving to scalable but precise filtering of documents for internet-scale systems. Accordingly, an automaton query storage device for storing said two-state input automaton (indexing queries).]

"an automaton-evaluating device for enabling a state transition by reading out two-state input automaton from said storage device and storing said automaton, while reading in said document and identifying said two states node, and performing a stream search by using states of a plurality of different types of nodes in the element identifiers included in the document and the query automaton and"[figure 2 and page 55 col. 1, XPATH is used to select entire documents rather than parts of documents. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Accordingly, an automaton-evaluating device (figure 2, filter engine) for enabling a state transition by reading out two-state input automaton from said storage device (figure 2, query index) and storing said automaton, while reading in said document (figure 2, xml document) and identifying said two states node, and performing a stream search (figure 3a and 3b, query) by using states of a plurality of different types of nodes (page 56, states of the XPATH queries) in the element identifiers (figure 3a and 3b, query identifier) included in the document (XML document)and the query automaton (query index)]

"a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node." [See page 57, column 2, 4th paragraph, All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List. The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph. If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not

the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)." Accordingly, a search result storage means for storing the output of the query automaton evaluator (A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List.), and thereafter outputting the stored output of the query automaton evaluator (candidate list) and the output of the searched node (candidate list)]

Altinel does not explicitly disclose, "a logical expression including a conjunction or a negative expression,".

On the other hand, W3C discloses on page 60, in addition to and- and or- expressions, Xpath provides a function named not that takes a general sequence as parameter and returns a Boolean value. Accordingly, a logical expression (logical expression) including a conjunction (AND) or a negative expression.

Both Altinel and W3C are within the same field of endeavor as Applicant's invention. It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to have applied the disclosure of W3C above to the disclosure of Altinel for the purpose of further evaluating and filtering documents.

Response to Arguments

8. Applicant's arguments with respect to claims 1, 3-4, 6, 10, 11, 14-15, 18-20, and 22 have been considered but are moot in view of the new ground(s) of rejection.

9. Applicant's assert the following directed towards the prior cited references (lettered):

A. Remarks page 21, Altinel fails to disclose "wherein the compiling device generates and registers a state transition by replacing an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state.

In response, this is moot on grounds of new rejection. Atinel discloses page 56 col. 2, Xfilter converts each XPATH query to a finite state machine. Further disclosing page 56 col. 2, the query index is built over the states of the XPATH queries. Further disclosing page 55 col. 1, how expressions such as //product [price/msrp < 300]/name are evaluated. Figure 3a and figure 3b disclose example queries and corresponding nodes as well as query index. Page 58 col. 1, discloses if it is not the final node of the query (i.e. final state) then the query is moved into its next state.

Accordingly, wherein the compiling device generates and registers a state transition by replacing an XPATH axis (Xfilter converts each XPATH query to a Finate State Machine) including an XPATH axis in the opposite direction (//product[price/msrp<300]/name], product)

and a logical expression (//product[price/msrp<300]/name], price/msrp<300) while keeping an input query expression equal in terms of search (query index is used to match documents to individual XPATH queries) , wherein the compiling device generates a query automaton (figure 3, query index) including a plurality of states (query index is built over the states of the XPATH queries) of the backward nodes (figure 3, Q1, a), a condition for transition (price/msrp<300), and at least a search state (if it is not the final node, the query is moved into its next state), is disclosed.

Altinel does not explicitly disclose, "a logical expression including a conjunction or a negative expression,".

On the other hand, W3C discloses under the logical expressions section on page 60, in addition to and- and or- expressions, Xpath provides a function named not that takes a general sequence as parameter and returns a boolean value. Accordingly, a logical expression (logical expression) including a conjunction (AND) or a negative expression.

Both Altinel and W3C are within the same field of endeavor as Applicant's invention. It would have been obvious to a person of an ordinary skill in the art at the time the invention was made to have applied the disclosure of W3C above to the disclosure of Altinel for the purpose of further evaluating and filtering documents.

B. Remarks page 21, Altinel fails to disclose said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node.

In response, figure 2, figure 3a, and page 58 of Altinel disclose the above asserted limitation. In particular, Altinel discloses said query automaton evaluator (figure 2, filter engine) determining a state transition of a node (states of the XPATH query) under determination by storing a left node (figure 3a, q1, a) and a lower node (figure 3a, q1, b) in correspondence with an identified element identifier (figure 3a, q1), and evaluating said query automaton with a search result of said left node and said lower node (pg. 58, if this is the final path node of the query then the document is deemed to match the query.).

For the reasons above, the limitations as claimed are still broad enough to read on the cited references.

C. Remarks page 22, Altinel teaches away from applicant's claimed invention by indexing queries and reversing queries and documents.

Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

D. Remarks pages 22-23, assertions directed towards Fernandez et. al.

In response, this is moot.

Conclusion

10. The prior art made of record listed on PTO-892 and not relied, if any, upon is considered pertinent to applicant's disclosure.

11. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Contact Information

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael D. Pham whose telephone number is (571)272-3924. The examiner can normally be reached on Monday - Friday 9am - 5:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John R. Cottingham can be reached on 571-272-7079. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/M. D. P./
Examiner, Art Unit 2167

/John R. Cottingham/
Supervisory Patent Examiner, Art Unit 2167